ASSEMBLER

Questo programma e' stato scritto appositamente per il Sinclair ZX Spectrum, e viene fornito in due versioni sulla stessa cassette per il 16K e per il 48K RAM; evidentemente, data la complessita' dell'argomento, non e' certo possibile insegnare a programmare in queste poche righe, pertanto vi consigliamo la lettura di testi specifici.
Nel nostro catalogo e' gla' da tempo disponibile un DISASSEMBLER totalmente combatibile con l'ASSEMBLER: se qualcuno volesse utilizzarli contemporaneamente sara' sufficiente caricare in memoria prima l'assemblatore e poi il disassemblatore.
Premendo un tasto a LOAD effetuato si cancellera' lo schermo, lasciando il programma inserito in memoria.

COME SCRIVERE PROGRAMMI IN LINGUAGGIO MACCHINA

Prima di tutto e' necessario creare uno spazio per il codice prodotto: il nostro consiglio e' di inserire un REM all'inizio del programma seguito da molteplici spazi; in questo modo, il primo indirizzo disponibile per accettare il linguaggio macchina sara' 23760. I normali mnemonici dello Z80 andranno scritti all'interno di REM e in caratteri minuscoli, esattamente come appaiono in fondo al manuale dello ZX Spectrum.
L'unica eccezione sono le istruzioni RET C,RET P,RET M e RET Z che devono sempre essere seguite da una virgola.
Inoltre, quando usate i salti condizionati, ricordatevi di inserire il segno (jr -3 o jr nc, +7).
I valori vanno sempre inseriti in DECIMALE.
I dati mnemonici devono essere preceduti da "REM go" e terminati con "REM finish".
Fatta eccezione per gli statements go e finish, che devono essere posti su linee separate, tutti gli altri statements possono essere posti su un'unica linea purche' separati dai due punti.
Digitando gli mnemonici bisoqna ricordare che la corretta posizione della virgola, dello spazio e delle parentesi e' estremamente importante. Consulta bene quindi il manuale dello Spectrum.

e pseudo istruzioni

Il programma contiene alcune istruzioni veramente utili che non sono ufficialmente contemplate nelle istruzioni dello Z80 ma che facilitano notevolmente la stesura dei programmi, esse sono:

ORG. Questa istruzioni (es.ORG.23760) indica all'assembler dove iniziare l'assemblaggio del programma. (Nell'esempio nel primo byte disponibile nel REM all'inizio del programma). Quindi, per ovvie ragioni, la prima istruzione che dovete dare e' appunto ORG.Non ci sono limitazioni al numero dei comandi ORG che potete impiegare al fine di saltare a nuovi indirizzi di assemblaggio. Se vi capita di dover lavorare in prossimita' dell'indirizzo 60.000 (o 30.000 per il 16K)puo' verificarsi che il vostro programma si sovrapponga all'assembler cancellandolo.
Questo ostacolo puo' essere aggirato con l'impiego di una forma speciale di ORG. Se impiegate ORG 23760 ma scritto in modo tale da funzionare correttamente quando verra' trasferito all'indirizzo 60.000. Dopo queta operazione basta eseguire SAVE "" CODE 23760,x quindi LOAD "" CODE 60.000,x per posizionare il programma all'indirizzo 60.000.

DEFB. Questo permette di forzare un byte di valore definito nell'indirizzo del momento. Puo' essere utile quando, si deve forzare un messaggio di errore con rst8. Per esempio rst8, defb1 fornisce il messaggio di errore "2 variable not found".

DEFW. Molto simile a defb ma con la differenza che puo' forzare una parola (2 bytes) piuttosto che 1 byte. L'argomento e' rappresentato da un numero compreso tra 0 e 65535. Il numero deve essere immesso nelle memorie con inizio dal bit meno significativo.

DEFS. Permette di inserire una stringa di caratteri ASCII nell'indirizzo attuale. Questo puo' essere utile per memorizzare del testo insieme al programma.

EQU. E' usato per specificare del titoli (Labels) e per collegare sezioni di codici tra loro (vedere di seguito).

LABELS

Possono essere utilizzati del titoli (Labels) per fare riferimento a determinate posizioni di memoria il cui indirizzo rimane sconosciuto, finche' il programma non e' completo. Le Labels possono avere lunghezze desiderate, ma con le seguenti restrizioni:

Il primo carattere deve essere maiuscolo.
Non devono essere usati il "+" e la parentesi destra ")"
Le Labels vengono scritte nel programma semplicemente come ogni altra istruzione.
L'esempio che segue mostra como puo' essere inserita l'etichetta "Elettronica CS":
ld a, (Elettronica CS); ld hl (Elettronica CS); ld hl. Elettronica CS; ld de, Elettronica CS; call Elettronica CS; jr nz, Elettronica CS; jr c, Elettronica CS eccetera.

Stesura di un codice automodificante

Supponete di avere la seguente porzione di codice:

.....Johnny; ld hl, (23700).

e di voler modificare l'indirizzo dalla posizione dove si trova hl. Potete in questo caso dare la seguente istruzione:

ld hl, 30.000; ld (Johnny +1), hl

Quando questa parte di programma viene eseguita verra' riscritto

....;Johnny; ld hl, (30.000;.....

Notare che in questo tipo di Label sono ammessi solo +1 e+2

Riferimento esplicito a Labels

A volte, specialmente con lo Spectrum 16 K, vi capitera' di scrivere un programma in codice macchina in varie sezioni riunendole poi in seguito .
Cio' puo' essere fatto se non vi riferite a Labels che siano al di fuori delle porzioni di codice che state scrivendo. Se lo fate le Labels non verranno rintracciate e vi si presentera' un messagglio,di errore.
Avete pero' la possibilita' di aggirare il problema impiegando l'istruzione EQU che fornisce all'assembler l'indirizzo della Label. Per esempio equ 23780 Fred fara' in modo che la Label "Fred" possa essere impiegata anche se l'indirizzo di quest'ultima non e' compreso nella porzione di codice che si sta assemblando.
Infine l'assembler posiziona i nomi e gli indirizzi delle Labels in una tabella. Lo spazio e' limitato e quando questa e' piena verra' indicato un codice di errore. Il numero delle Labels che si possono impiegare dipende dalla loro lunghezza. Ne potete scrivere anche 100 da 5 caratteri.

Commenti

I commenti possono essere inseriti nel programma facendoli precedere da un punto esclamativo. I primi 32 carattteri di un commento appaiono sollo schermo o sulla stampante. Se desiderate dei commenti piu' estesi, scrivetene diversi consecutivi.

Esecuzione del programma assembler

La versione 48K va esequita tramite RANDOMIZE USR 60.000 (RANDOMIZE USR 27500 per la versione 16 K).Questa istruzione puo' essere inclusa in una riga in modo che il programma puo' essere fatto partire con RUN. In questo caso il programma puo' apparire come nell'esempio sottoriportato:

```
10 REM Spazio per il codice macchina
20 REM go
30 REM org 23760 (54321); equ 30000 Altri
40 REM !inizio del I^ listato
50 REM Start;ld hl, (Altri);inc hl;ld a, (hl);cp 240; jr z, -6
60 REM sub 34; ret z,;ld hl, (Start +1); dec hl
70 REM jr Start
80 REM !inizio del II^ listato
90 REM ORG 21000
100 REM defs Questo e' testo;ld hl,31000;ld a, (hl);cp 84
110 REM ret nz;rst 8; defs 10
120 REM !Fine delle due sezioni
130 REM finish
140 RANDOMIZE USR 60000 o RANDOMIZE USR 27500
```

Quando viene eseguito, l'indirrizo assemblaggio di ogni istruzione, il suo codice esadecimale e i primi 13 caratteri del mnemonico appaiono sullo schermo ogni istruzione viene assemblata 2 volte in modo da calcolare l'indirizzo di destinazione del prossimo salto. L'assemblaggio si arresta quando lo schermo e' completo.
Premendo il tasto "SPACE" l'assemblaggio si arresta. Premendo "p" durante il secondo passo, tutto il listato viene inviato alla stampante e premendo ogni altro tasto continua l'assemblaggio.

Errori

Se durante l'assemblaggio viene rilevato un errore, l'operazione si arresta con due messaggi di errore. Il primo e' intermittente e indica il numero di linea, l'instruzione e il tipo di errore nell'istruzione.
Il secondo messaggio e' uno di quelli normalmente indicati dal Sinclair.

Ci sono tre possibilita':

1) Se un numero e' maggiore di 255 (per un singolo byte) o e' maggiore di 65535 (per un doppio byte) l'assembler sottrae ripetutamente 256 o 65536 finche' ottiene un numero valido. Se non si raggiunge un risultato valido compare l'errore "B Integer out of range"

2) Se fate riferimento ad una Label inesistente o se avete completato la tavola delle Labels vi si presenta l'errore "2 Variable not found".

3) Tutti gli altri errori (come errori di sintassi) si presenteranno con l'indicazione "Q Parameter error".
Ci sono casi in cui l'assembler corregge automaticamente i vari errori; come ad esempio nel caso di ld r , 8 che verra' corretto in ld r, a.